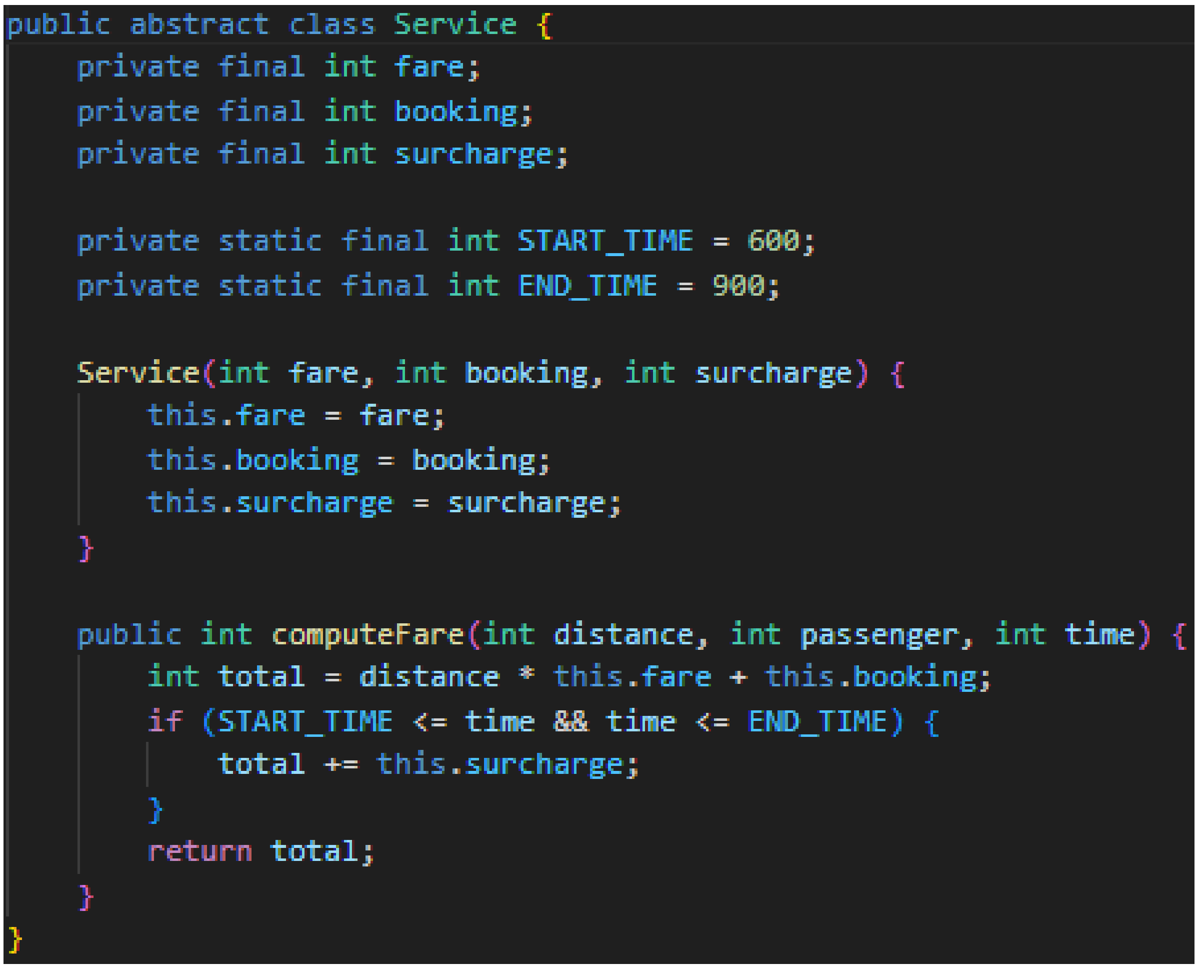
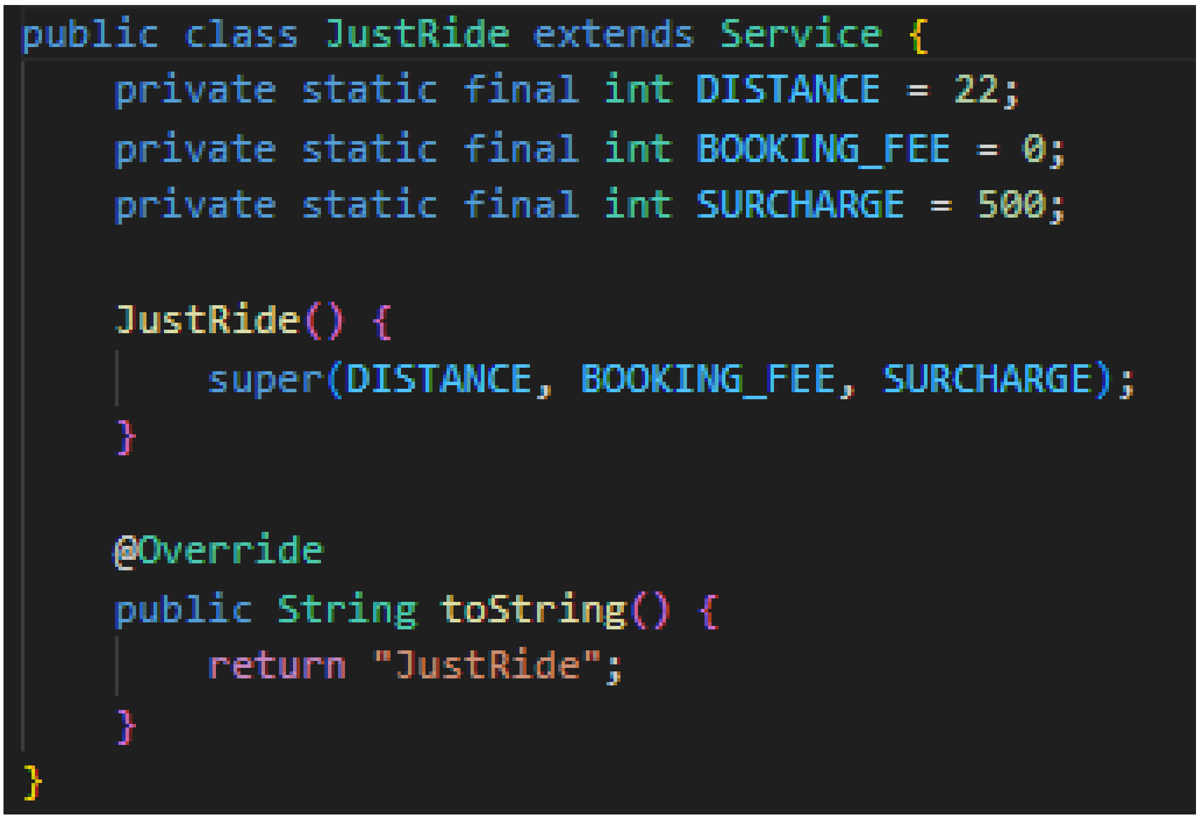
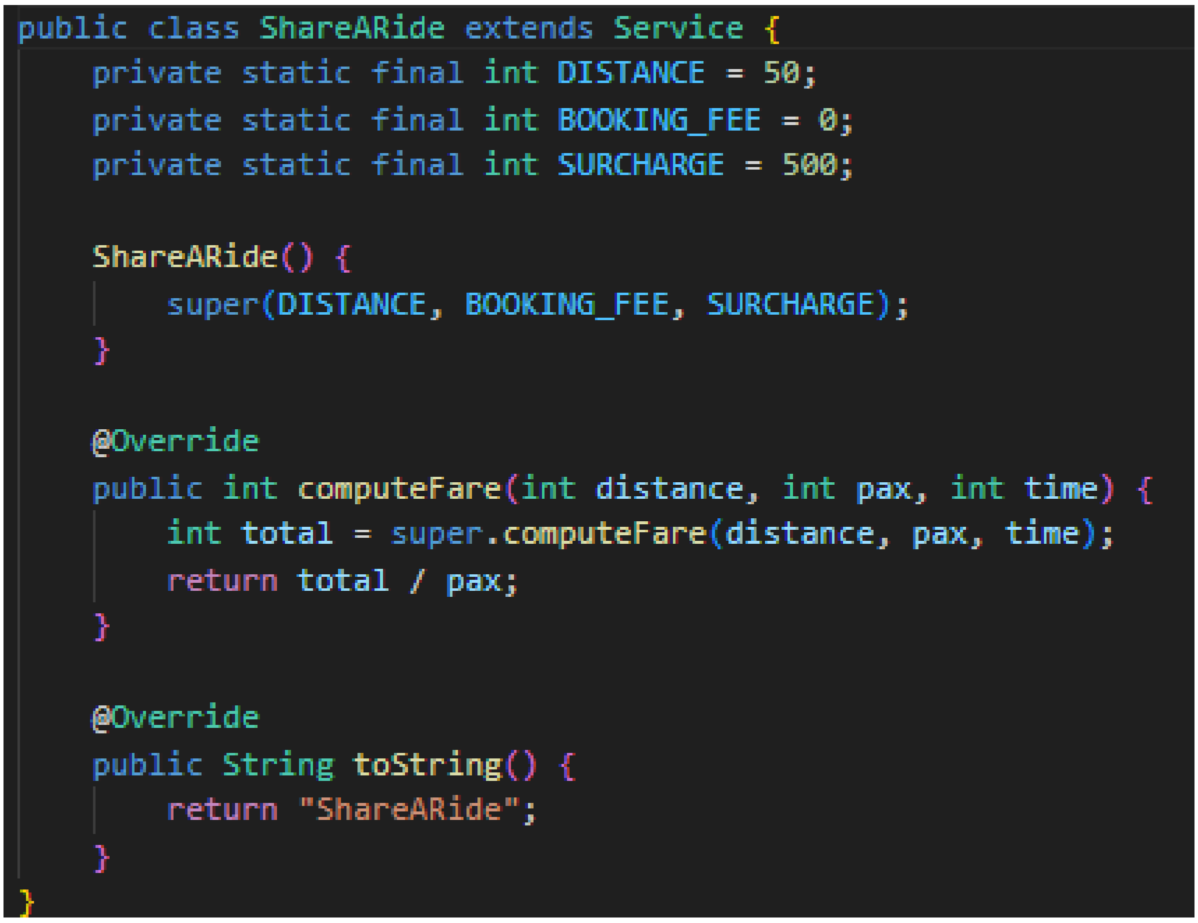


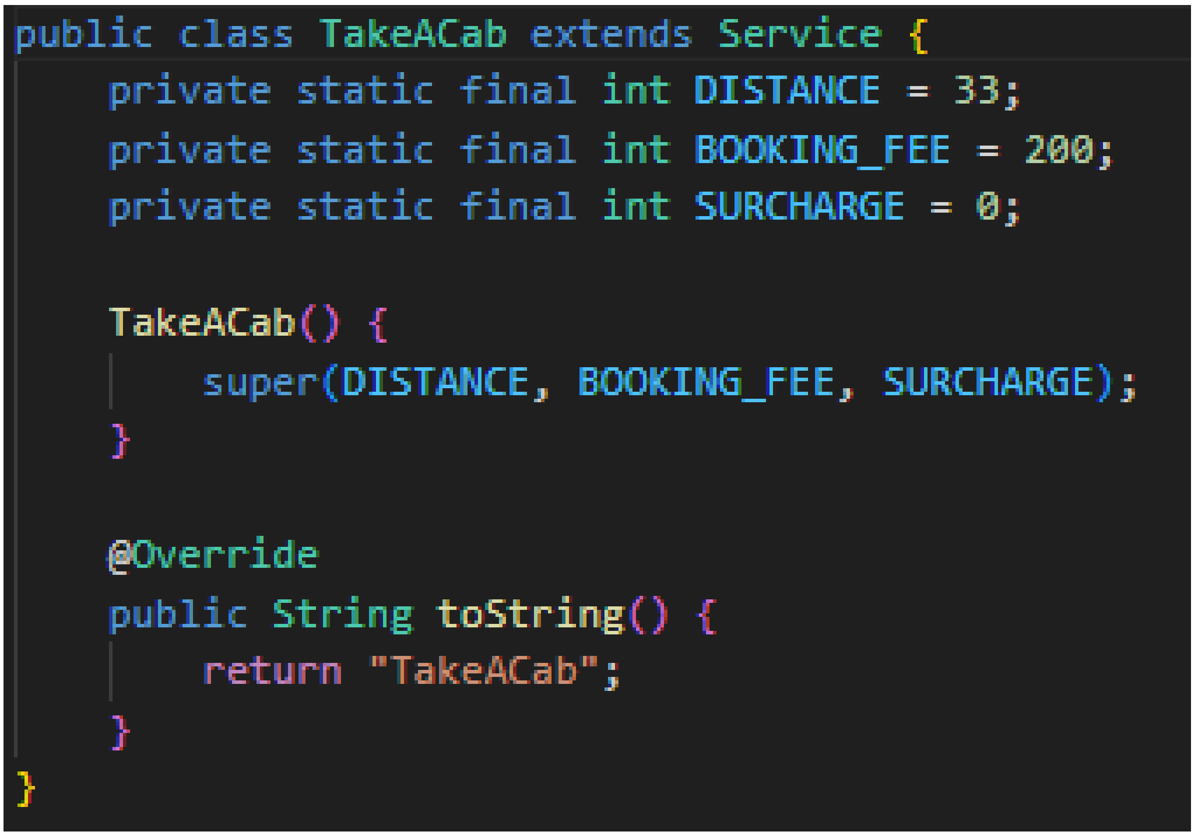
```java
public abstract class Service {
    private final int fare;
    private final int booking;
    private final int surcharge;

    private static final int START_TIME = 600;
    private static final int END_TIME = 900;

    Service(int fare, int booking, int surcharge) {
        this.fare = fare;
        this.booking = booking;
        this.surcharge = surcharge;
    }

    public int computeFare(int distance, int passenger, int time) {
        int total = distance * this.fare + this.booking;
        if (START_TIME <= time && time <= END_TIME) {
            total += this.surcharge;
        }
        return total;
    }
}
```



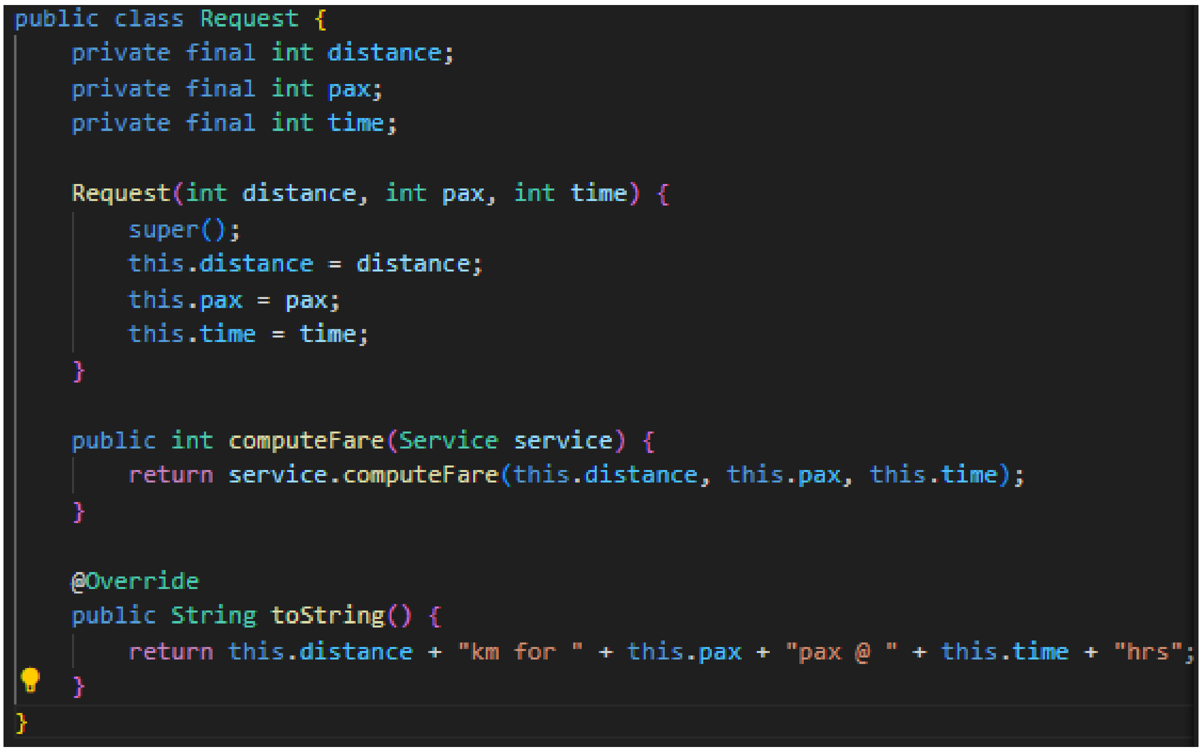
```java
public class JustRide extends Service {
    private static final int DISTANCE = 22;
    private static final int BOOKING_FEE = 0;
    private static final int SURCHARGE = 500;

    JustRide() {
        super(DISTANCE, BOOKING_FEE, SURCHARGE);
    }

    @Override
    public String toString() {
        return "JustRide";
    }
}
```



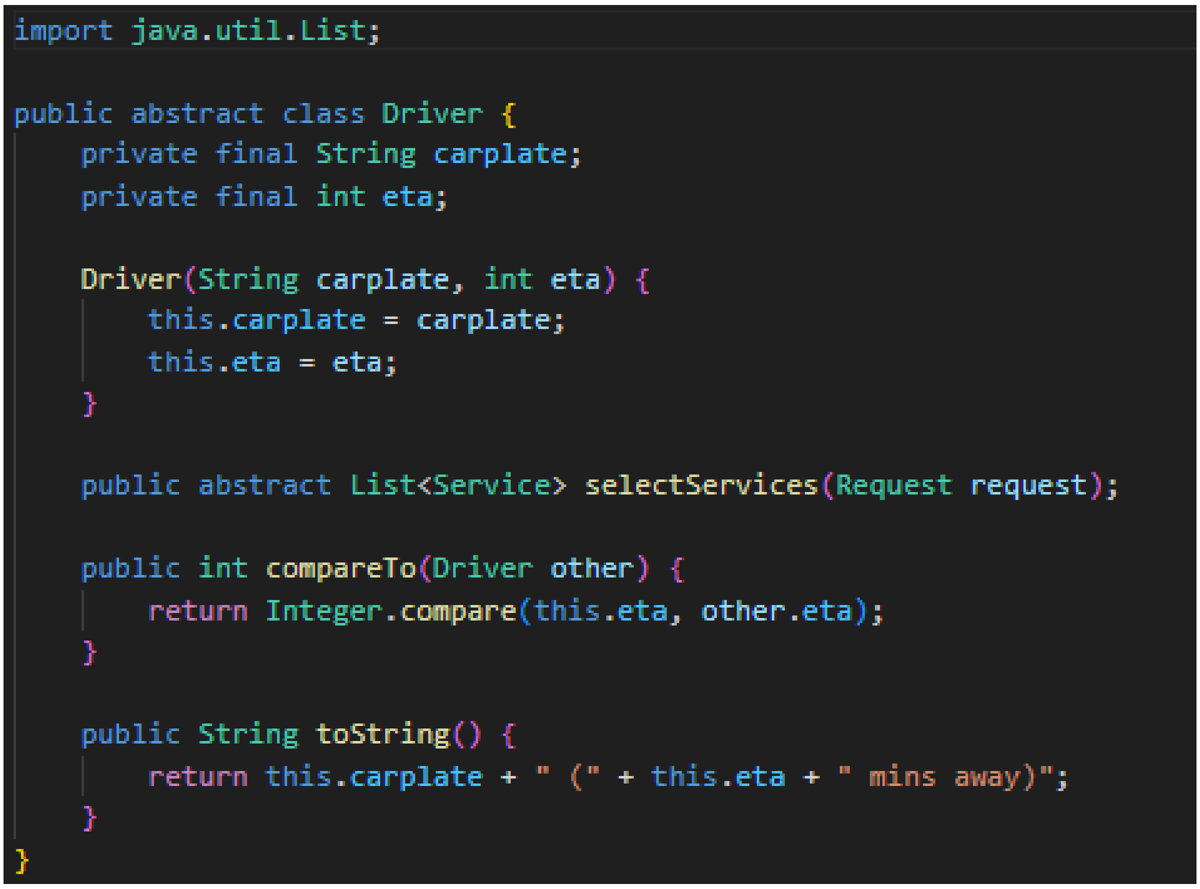
```java
public class ShareARide extends Service {
    private static final int DISTANCE = 50;
    private static final int BOOKING_FEE = 0;
    private static final int SURCHARGE = 500;

    ShareARide() {
        super(DISTANCE, BOOKING_FEE, SURCHARGE);
    }

    @Override
    public int computeFare(int distance, int pax, int time) {
        int total = super.computeFare(distance, pax, time);
        return total / pax;
    }

    @Override
    public String toString() {
        return "ShareARide";
    }
}
```



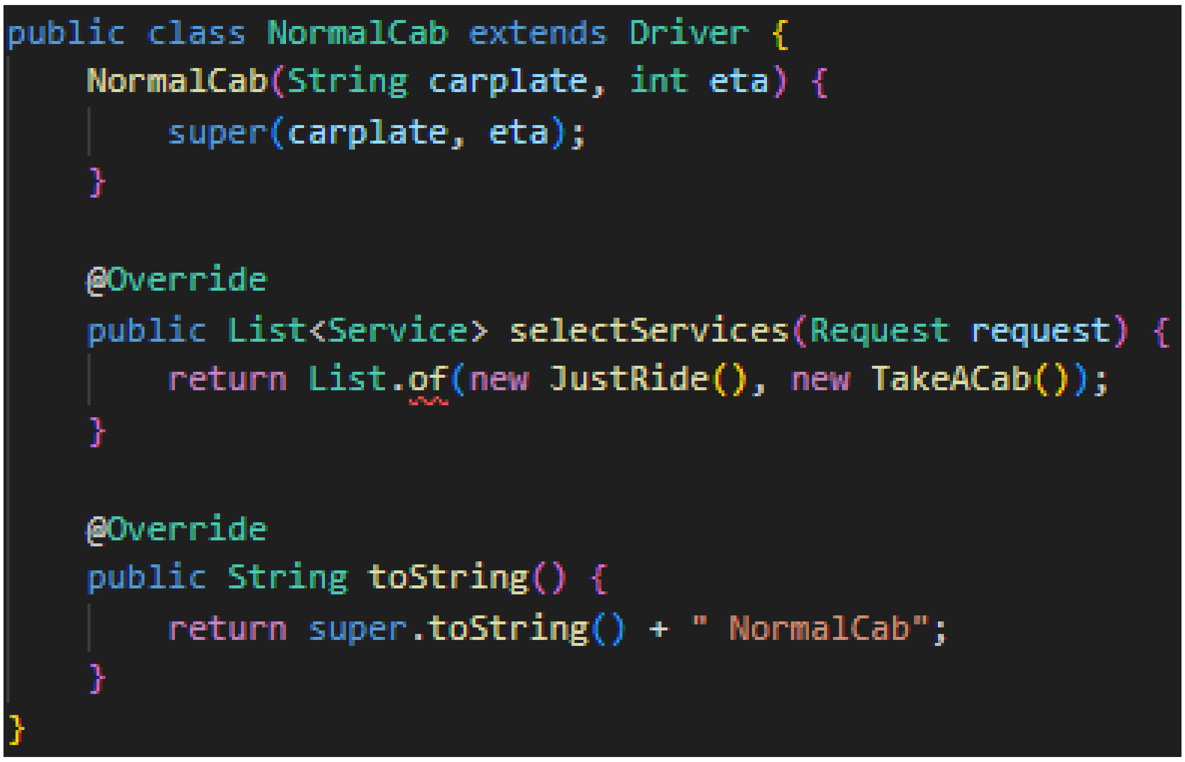
```java
public class TakeACab extends Service {
    private static final int DISTANCE = 33;
    private static final int BOOKING_FEE = 200;
    private static final int SURCHARGE = 0;

    TakeACab() {
        super(DISTANCE, BOOKING_FEE, SURCHARGE);
    }

    @Override
    public String toString() {
        return "TakeACab";
    }
}
```



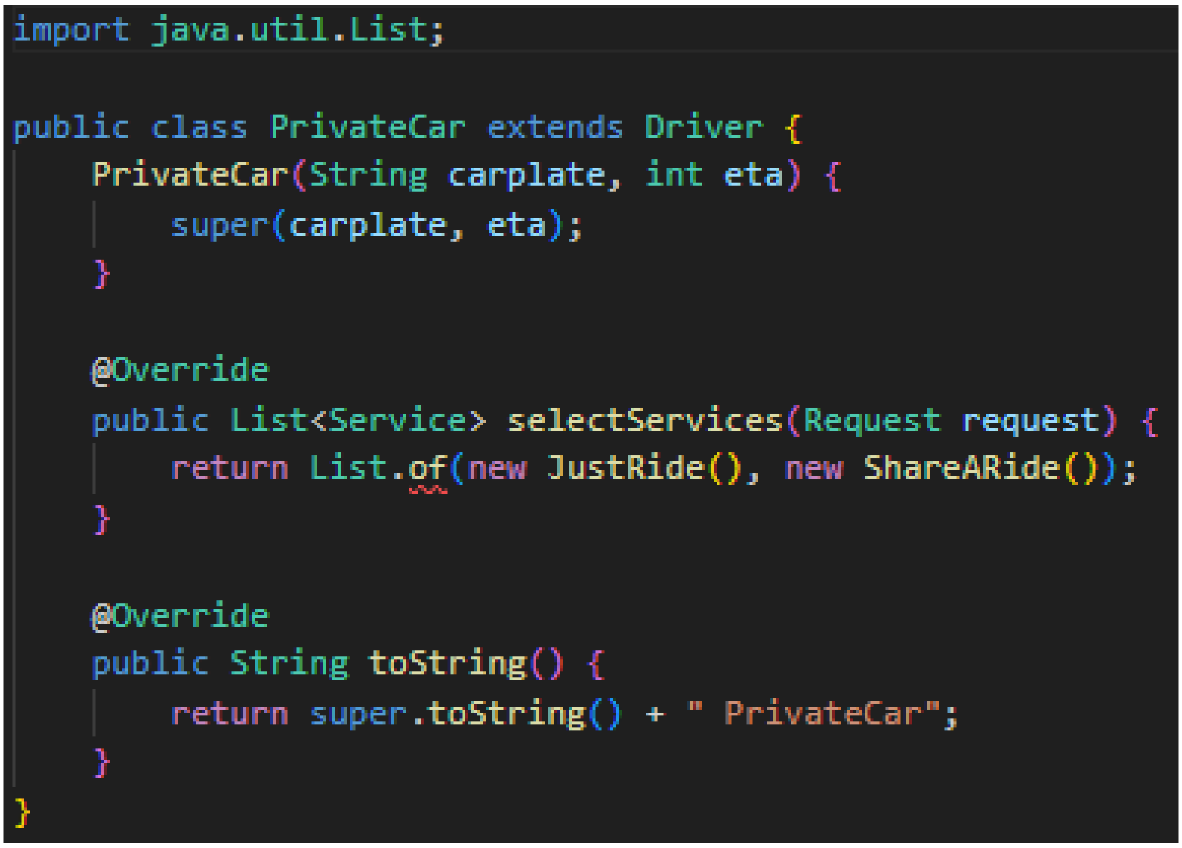
```java
public class Request {
    private final int distance;
    private final int pax;
    private final int time;

    Request(int distance, int pax, int time) {
        super();
        this.distance = distance;
        this.pax = pax;
        this.time = time;
    }

    public int computeFare(Service service) {
        return service.computeFare(this.distance, this.pax, this.time);
    }

    @Override
    public String toString() {
        return this.distance + "km for " + this.pax + "pax @ " + this.time + "hrs";
    }
}
```



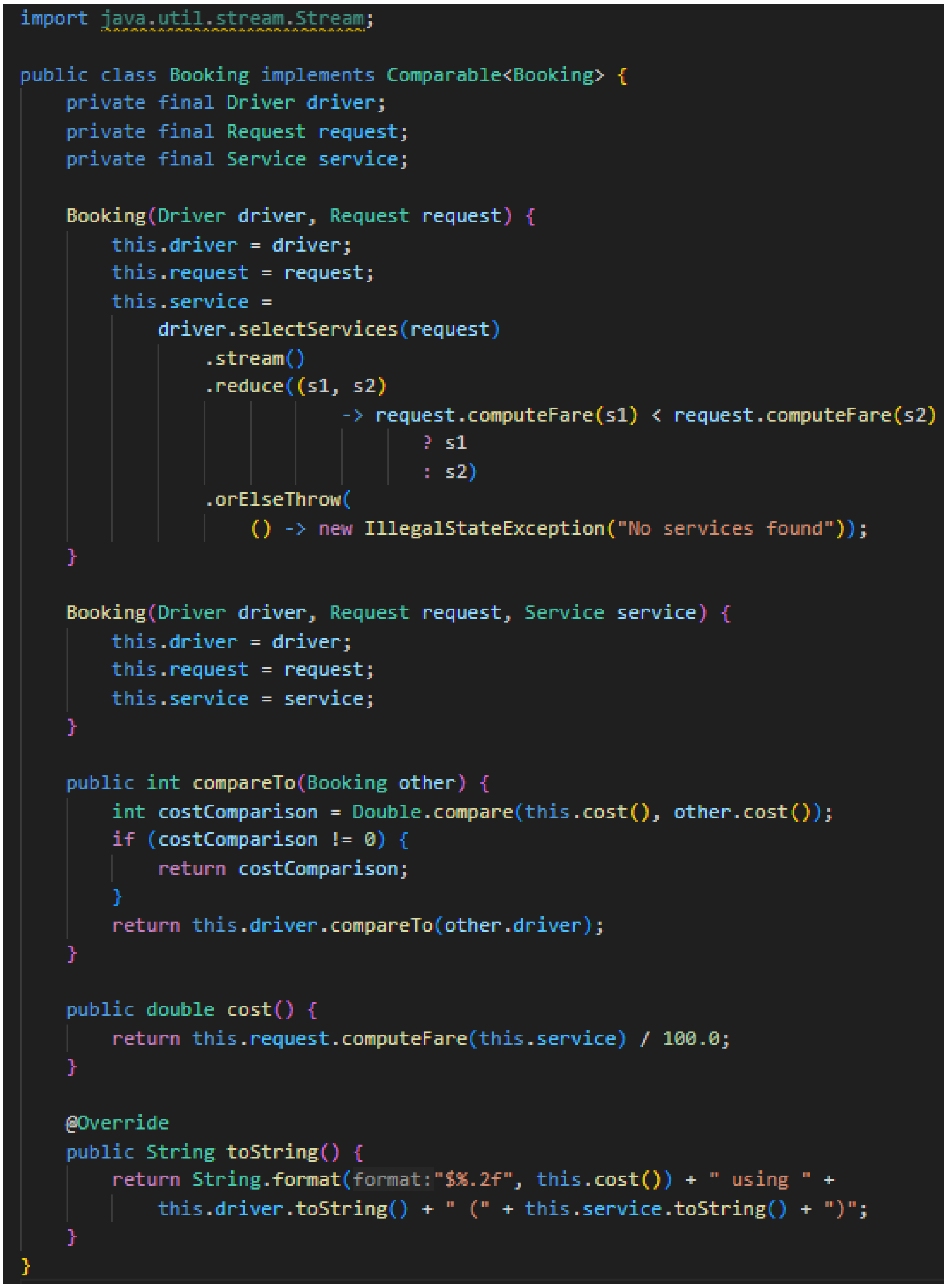
```java
import java.util.List;

public abstract class Driver {
    private final String carplate;
    private final int eta;

    Driver(String carplate, int eta) {
        this.carplate = carplate;
        this.eta = eta;
    }

    public abstract List<Service> selectServices(Request request);

    public int compareTo(Driver other) {
        return Integer.compare(this.eta, other.eta);
    }

    public String toString() {
        return this.carplate + " (" + this.eta + " mins away)";
    }
}
```



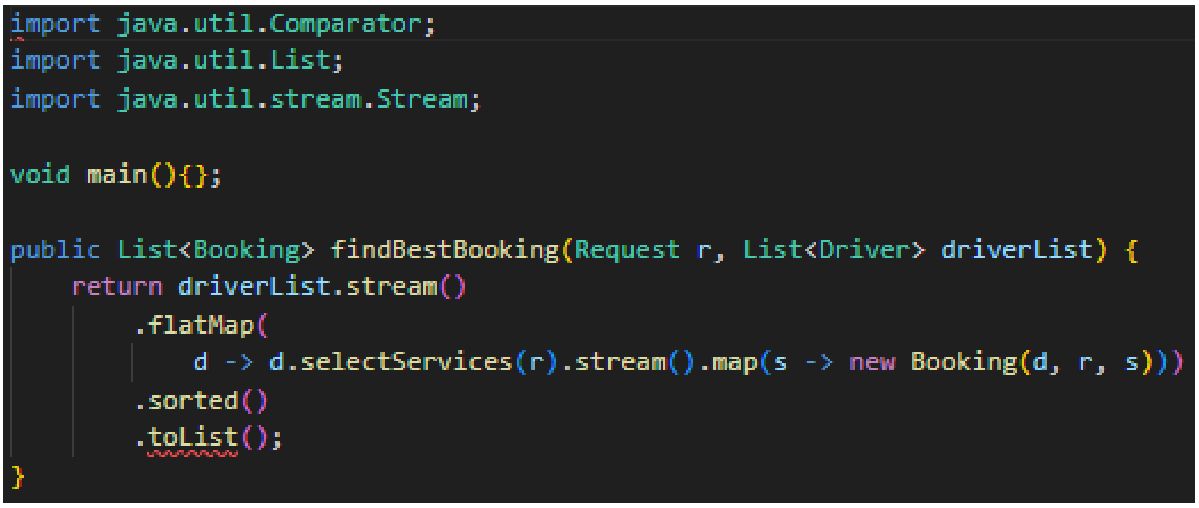
```java
public class NormalCab extends Driver {
    NormalCab(String carplate, int eta) {
        super(carplate, eta);
    }

    @Override
    public List<Service> selectServices(Request request) {
        return List.of(new JustRide(), new TakeACab());
    }

    @Override
    public String toString() {
        return super.toString() + " NormalCab";
    }
}
```



```java
import java.util.List;

public class PrivateCar extends Driver {
    PrivateCar(String carplate, int eta) {
        super(carplate, eta);
    }

    @Override
    public List<Service> selectServices(Request request) {
        return List.of(new JustRide(), new ShareARide());
    }

    @Override
    public String toString() {
        return super.toString() + " PrivateCar";
    }
}
```



```java
import java.util.stream.Stream;

public class Booking implements Comparable<Booking> {
    private final Driver driver;
    private final Request request;
    private final Service service;

    Booking(Driver driver, Request request) {
        this.driver = driver;
        this.request = request;
        this.service =
            driver.selectServices(request)
                .stream()
                .reduce((s1, s2)
                            -> request.computeFare(s1) < request.computeFare(s2)
                                ? s1
                                : s2)
                .orElseThrow(
                    () -> new IllegalStateException("No services found"));
    }

    Booking(Driver driver, Request request, Service service) {
        this.driver = driver;
        this.request = request;
        this.service = service;
    }

    public int compareTo(Booking other) {
        int costComparison = Double.compare(this.cost(), other.cost());
        if (costComparison != 0) {
            return costComparison;
        }
        return this.driver.compareTo(other.driver);
    }

    public double cost() {
        return this.request.computeFare(this.service) / 100.0;
    }

    @Override
    public String toString() {
        return String.format(format:"$%.2f", this.cost()) + " using " +
            this.driver.toString() + " (" + this.service.toString() + ")";
    }
}
```


```java
import java.util.Comparator;
import java.util.List;
import java.util.stream.Stream;

void main(){};

public List<Booking> findBestBooking(Request r, List<Driver> driverList) {
    return driverList.stream()
        .flatMap(
            d -> d.selectServices(r).stream().map(s -> new Booking(d, r, s)))
        .sorted()
        .toList();
}
```